一、 DAY01数据库基本概念

- 学习目标：
    - 了解数据库的基本概念
    - 掌握数据库的基本查询语句
    - 了解数据库的连接查询语句

1、初识数据库

1.1、数据库所处的位置

- Web开发
- 客户端(页面：展示，数据！)
- 服务端(连接点：连接前端(控制页面跳转，和给前端传递数据)，连接数据库)
- 数据库是在服务器里面，服务器可能是 LINUX(centos7, ubuntu, redhat), UNIX, MACOS

1.2、为什么学习数据库

1. 被迫需求：存数据
2. 大数据时代 用户画像
3. 岗位需求
4. 数据库是所有软件体系中最核心的存在(DBA database administrator)

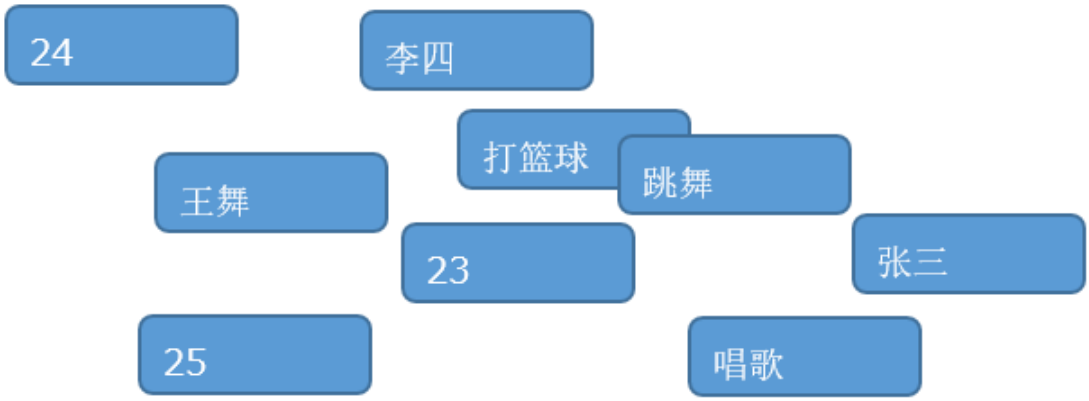
1.3、什么是数据库(DB，DataBase)

- 概念： 数据库，即存储在磁带、**磁盘**、光盘或其他**外存介质**上、按一定结构组织在一起的相关数据的集合。
    - 随着科技的发展存储介质的读取速度越来越快
- 按照**数据结构**来**存储**和**管理数据的**仓库，数据在硬盘上
- 作用：存储数据，管理数据
- 数据库下面可以没有表, 也可以有一张或者多张表

- 现实世界使用数据库的场景：
    - 银行账户
    - 电话账单
    - 游戏账户信息

1.4、填填看

姓名	年龄	爱好



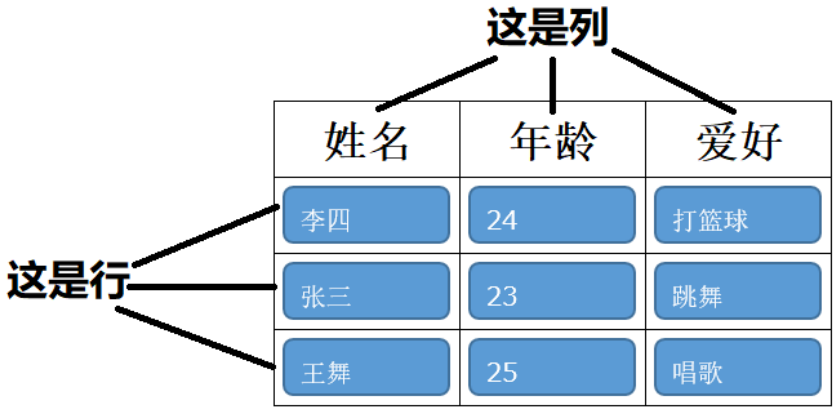
1.5、软件测试工程师使用数据库来做什么，需要注意什么?

1. 执行测试用例时，有时需要到数据库验证数据的准确性与完整性
2. 进行bug定位时，有时需要到数据库查看数据的详细信息
3. 构造某种测试场景时，可以在数据库里直接添加数据，要比使用界面更有效率
4. **不要改动数据库里面的数据**

1.6、数据库分类

关系型数据库：(MySQL)

- **MySQL**，Oracle，Sqlite，PostgreSql，MS SQL Server
- 通过表和表之间，行和列之间的关系进行数据的存储
    - **列**是存储在表中的一块数据(**也就是表中的表头**)
    - **行**是一组能够描述某个事物的列的集合(**也就是表头下面对应的数据**)

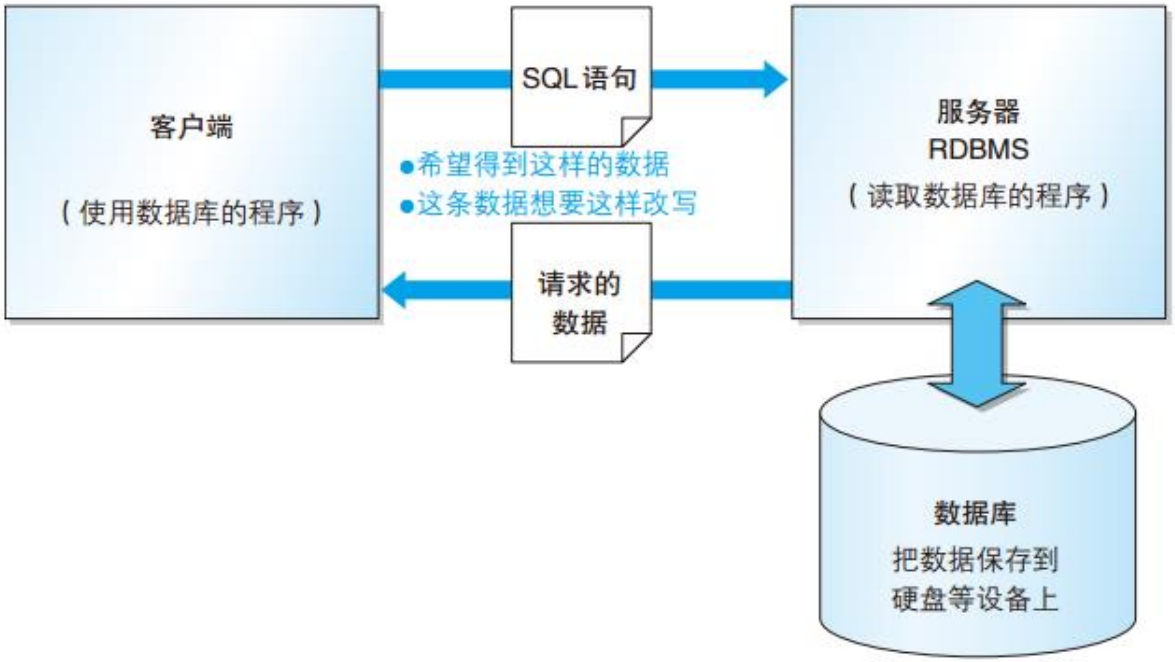


非关系型数据库：(NoSQL)Not Only

- **Redis**，MongoDB　redis是存放在内存的数据库，断电不会丢失数据
- 非关系型数据库，对象存储，通过对象的自身的属性来决定
  - 键(key)：值(value)

1.7、RDBMS(关系型数据库管理系统)Relational Database Management System

- 管理数据库的软件，用来科学有效的**管理**，**维护和获取数据**
- MySQL就是**数据库管理系统**
- 数据库的客户端，服务端以及服务器RDBMS的关系



1.8、Mysql是什么

- **概念：**是现在**流行的，开源的，关系型**数据库管理系统
- **历史：**由瑞典MySQL AB 公司开发，目前属于 Oracle 旗下产品
- **特点：**
  - 小巧, 功能齐全

- 使用便捷
- **跨平台：**可运行于Windows或Linux操作系统
- **官网：**https://www.mysql.com/

1.9、SQL(Structure Query Language，结构化查询语言)是用来操作关系数据库的语言(了解)

- SQL语法注意事项:
  1. 不区分大小写，建议大写
  2. 语句可以换行，单词不能换行，语句请一定以;结尾
  3. --空格 注释
  4. 蓝色 关键词(不可修改)
  5. 黑色 普通文本(自定义)
  6. 数值型数据不需要引号，字符串必须加引号
  7. SQL语句中所有的标点符号都必须要是**英文的！！！**
- 浅谈**SQL语法规范**:
  1. 用单引号将字符串括起来
  2. 字段别名用as, 表别名不用as
  3. 逻辑运算符前后用空格隔开
  4. SQL语句尽量写多行
  5. 语句的末尾加上英文的分号(;)
  6. 创建数据表的时候, 表名和字段名用反撇号(`)括起来

2、数据库操作浅析

2.1、结构化查询语言分类

名称	解释	命令
DDL(Database Definition Language)数据库定义语言	定义和管理数据对象，如数据库，数据表等	CREATE, DROP, ALTER
DML(Database Manipulation Language)数据库操作语言	用于操作数据库对象中所包含的数据	INSERT, UPDATE, DELETE
DQL(Database Query Language)数据查询语言	用于查询数据库数据	SELECT
DCL(Data Control Language)数据控制语言	用于管理数据库的语言，包括管理权限及数据更改	GRANT, commit, rollback

2.2、数据库常见对象

- 数据库(Database)

  同比为某律师事务所的档案仓库(各种案子的文档汇总在不同的文档架上)

- 表(Table)

  同比为档案仓库中的文档架

- 字段|列(Column)

  同比为档案仓库中的文档夹上的一个个文档(编号、标题、入库时间、入库人.....)

2.3、数据库操作(了解)

- 打开Navicat.exe文件 ---> 点击连接 ---> 点击数据库 ---> Navicat点击查询--->新建查询
- 给数据库命名的时候要具有描述性

```
-- 创建一个叫做school的数据库
CREATE DATABASE school;

-- 删除数据库
DROP DATABASE school;

-- 展示当前所有的数据库
SHOW DATABASES;

-- 选择数据库，操作指定数据库之前，必须先要选择指定的数据库(命令行)
USE school;
```

- **好好敲一敲上面的代码，并且一定注意SQL语句中所有的标点符号都必须要是英文的！！！**
- 敲代码遇到报错时的排错小技巧:
  - 检查自己在哪个数据库里面
  - 检查当前数据库里有哪些表(SHOW TALBES;)
  - 检查自己是否无意中敲了中文标点符号(比如中文的逗号分号等)

2.4、数据库表的列类型(了解)

> **数值 (如：1, 2, 3.14)**
>
> - tinyint　　十分小的数据　1个字节
> - smallint　　较小的数据　2个字节
> - mediumint　　中等大小的数据　3个字节
> - **int　　标准的整数　　4个字节**
> - big　　较大的数据　　8个字节
> - float　　浮点数　　4个字节
> - **double**　浮点数　　小数　　8个字节(精度问题)
> - **decimal**　字符串形式的浮点数　(金融计算的时候一般使用)

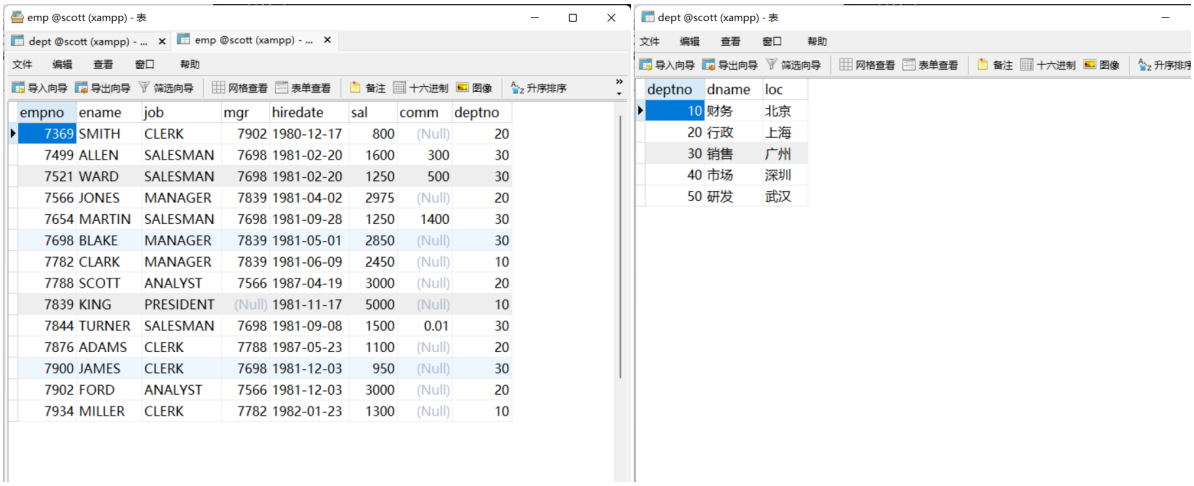
> **字符串**
>
> - **char**　　**固定大小字符串　0-255**　char(5)　00001
> - **varchar**　**可变字符串**　　**0-65535**　varchar(5)　1
> - tinytext　微型文本　　2^8 - 1
> - text　　文本串　　2^16 -1　保存大文本

> **时间日期**
>
> - date　　YYYY-MM-DD，日期 (year,month,day)
> - time　　HH: MM: SS　时间格式(hour:minutes,seconds)
> - **datetime　YYYY-MM-DD HH: MM：SS (年月日，时分秒)最常用的时间格式**
> - year　年份

> > **null**
> >
> > - 没有值，未知
> > - 注意，不要使用null进行运算，结果肯定为null

2.5、执行sql文件(导入教学数据库资料)

1. 右击已经新建的连接
2. 点击运行sql文件
3. 点击..按钮去查找sql文件的路径
4. 找到保存在本地的《教学数据备份.sql》文件选中打开
5. 点击开始按钮
6. 看到**successfully**表示执行sql文件成功
7. 点击关闭回到navicat的主界面
8. 右击已经新建的连接，刷新(如果能看到**scott**数据库则说明数据成功导入)



1. SMITH在哪里工作？

2. SMITH的领导叫什么名字？
3. 销售部门中岗位是CLERK的员工叫什么名字？
4. KING有哪几个直接下属？
5. MILLER同部门的同事中，哪些人的工资比MILLER高？

3、数据库操作实践 - DQL(Database Query Language)数据库查询语句(最重点)

3.1、Select完整的语法

```
select [all | distinct] <select_expr>, <select_expr>, ...   --5
from <table_reference>                                      --1
[where <where_condition>]                                   --2
[group by <col_list>]                                       --3
[having <having_condition>]                                 --4
[order by <order_condition>]                                --6
[limit <number>]                                            --7
```

3.2、基础查询

- SELECT **过滤出满足条件的列**
- FROM **确定要查询的数据来自哪张表**
- WHERE **过滤出满足条件的数据行**

```
-- 查询所有的员工 SELECT 字段 FROM 表
select *
from scott.emp;

-- 查询全部的部门,数据库 + 表名
select *
from scott.dept;

-- 查询指定字段，员工号码和员工名字
select empno, ename
from scott.emp;

-- 别名，给结果起一个名字 AS，可以给字段取别名，也可以给表取别名
-- as可以加也可以不加
-- 给ename起别名为员工姓名，empno起别名为员工编号，job起别名为岗位
-- select ename as '员工姓名', empno '员工编号', job 岗位
select ename as '员工姓名', empno as '员工编号', job as '岗位'
from scott.emp;
```

语法:SELECT 字段，...FROM 表

> 去重 distinct

作用：去除SELECT查询出来的结果中重复的数据，只显示一条

```sql
-- 查询有哪些岗位
select job
from scott.emp;

-- 查询去除重复岗位后的结果
select distinct job
from scott.emp;
```

> 数据库的列(表达式)(了解)

```sql
-- 查询系统版本
select version();
```

3.3、where条件字句

作用： 检索数据中**符合条件**的值

搜索的条件由一个或者多个表达式组成

> 逻辑运算符1

运算符	语法	描述
>	a > b 2>3	左边是否大于右边，是为真，不是为假
>=	a >= b 3>=3	左边是否大于等于右边，是为真，不是为假
<	b < a 1<2	左边是否小于右边，是为真，不是为假
<=	b <= a 2<=1	左边是否大于等于右边，是为真，不是为假
=	a = c 1=1	左边和右边是否相等，是为真，不是为假
!=	a != c 1!=2	左边和右边是否不相等，是为真，不是为假

> 逻辑运算符2

运算符	语法	描述
and &&	a and b a && b 2=1 and 2>1	逻辑与，两个都为真，结果为真
or ||	a or b a || b 2=2 or 1>2	逻辑或 ，一个为真，结果为真
not !	not a ! a !true !false	逻辑非， 真为假，假为真

```sql
-- ----------------------  where ------------------------
-- 查询部门10的员工信息(员工编号，员工姓名)
SELECT empno, ename
FROM scott.emp
WHERE deptno = 10;

-- 查询不是部门10的员工信息(员工编号，员工姓名)
SELECT empno, ename
FROM scott.emp
```

```
WHERE deptno != 10;

-- 工资不是5000的员工
SELECT *
FROM scott.emp
WHERE sal != 5000;

-- 查询岗位是MANAGER的员工编号、员工姓名、岗位、部门编号、工资
SELECT empno, ename, job, deptno, sal
FROM scott.emp
WHERE job = 'MANAGER';

-- 查询工资在800到2000之间的员工
SELECT *
FROM scott.emp
WHERE sal >= 800 and sal <= 2000;

-- 查询工资为5000或1300然后岗位为CLERK的员工
SELECT *
FROM scott.emp
WHERE (sal = 5000 or sal = 1300) and job = 'CLERK';
```

> 模糊查询：比较运算符

运算符	语法	描述
IS NULL	a is null	如果操作符为NULL，则结果为真
IS NOT NULL	a is not null	如果操作符不为 NULL, 则结果为真
BETWEEN AND	a between b and c	若 a 在 b 和 c 之间，则结果为真
like	a like b	SQL匹配，如果a匹配b，则结果为真(无法确切知道所要查找的值，而只知道所要查找的数据符合的模式时)
in	a in (a1,a2,a3...)	假设a在a1,或者a2...其中的某一个值中，结果为真(确切知道所要查找的内容，且为多个值时)

```
-- --------null / not null---------
-- 查询没有奖金的员工信息
SELECT *
FROM scott.emp WHERE comm is null;

-- 查询有奖金的员工信息
SELECT *
FROM scott.emp WHERE comm is not null;

-- 区间查询，查询工资在800-2000之间的员工信息(包含)
select *
from scott.emp
where sal between 800 and 2000;

-- -------------------- 模糊查询 like --------------------
```

```sql
-- like结合 %(代表0到任意个字符) _(一个字符)
-- 查询S开头的员工姓名
select ename
from scott.emp
where ename like 's%';
-- where binary ename  like 's%';

-- 查询结尾是MITH开头仅一个字的员工
select ename
from scott.emp
where ename like '_MITH';


-- 查询姓名是A开头的员工信息 或者 工资小于1000的员工信息信息
select *
from scott.emp
where ename like 'a%' or sal < 1000;

-- 查询姓名中有N的员工信息
SELECT *
FROM scott.emp
WHERE ename like '%N%';

-- ----------------------- in(具体的一个或者多个值) ------------------------
-- 查询员工编号是7698,7782,7788,7839的员工信息
SELECT *
FROM scott.emp
WHERE empno in(7698,7782,7788,7839);

-- 查询员工姓名是BLAKE,CLARK,SCOTT,KING的员工信息
SELECT *
FROM scott.emp
WHERE ename in('BLAKE','CLARK','SCOTT','KING');
```

3.4、练习题

```sql
-- 练习题
-- 查询没有奖金并且工资在1100-1600之间的员工信息
-- 1, 没有奖金
-- 2, 工资在1100-1600之间
-- 3, select *

-- 查询员工编号7788,7566,7369,7698中，哪个员工的姓名倒数第2个字母是E


-- 查询30部门中名字由5个字母组成并且名字中没有M的员工信息


-- 查询10部门所有的员工信息和30部门中工资在1100-1600的员工信息


-- 查询10部门或者20部门中并且岗位是CLERK的员工信息


-- 查询不是10部门的员工


```

```
-- 查询除去以下条件的员工以外，公司的其它员工。
-- 条件：10部门所有的员工信息或者30部门中工资在1100-1600的员工信息
```

3.5、连接查询

- 连接两个或者两个表以上的查询
  - 超过三张表**性能**会有问题
- 目的
  - 查询的字段不在一个表里，但是需求是查询并显示**多张表里的字段**所以需要用到连接查询

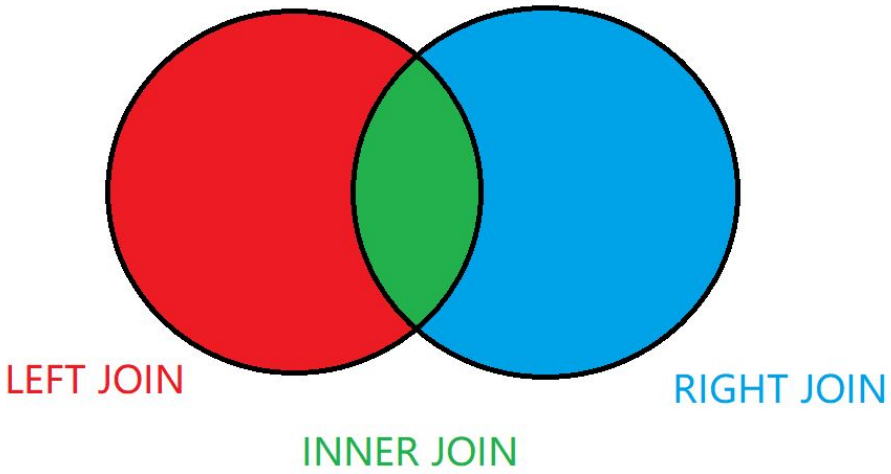
> 连接查询试一试

```
select *
from user_tbl, sal_tbl;
-- 笛卡尔积
-- 假设集合A={a, b}，集合B={0, 1, 2}，
-- 则两个集合的笛卡尔积为
-- {(a, 0), (a, 1), (a, 2), (b, 0), (b, 1), (b, 2)}

select *
from user_tbl, sal_tbl
where user_tbl.user_id = sal_tbl.user_id;
```

> JOIN

- 内连接 INNER JOIN / JOIN
- 外连接 LEFT JOIN 和 RIGHT JOIN




```
-- 连表查询
-- 查询员工的部门名称(员工编号，员工姓名，员工工种，员工部门简称)
-- inner join
SELECT e.empno, e.ename, e.job, d.dname
FROM scott.emp e
INNER JOIN scott.dept d
ON e.deptno=d.deptno;
```

```sql
-- left join, 查询没有员工的部门
SELECT e.*,d.*
FROM scott.dept e
LEFT JOIN scott.emp d
ON e.deptno=d.deptno
WHERE e.deptno is NULL;

-- right join, 查询没有员工的部门
SELECT e.*,d.*
FROM scott.emp e
RIGHT JOIN scott.dept d
ON e.deptno=d.deptno
WHERE e.deptno is NULL;
```

操作	描述	
inner join	表中至少有一行匹配，就返回	
left join	会从左表中返回所有的值，即使右表中没有匹配，显示为null	
right join	会从右表中返回所有的值，即使左表中没有匹配，显示为null	

3.6、SQL语句对齐工具

https://poorsql.com/

二、Day02 Mysql提升

- 学习目标
  - DDL 数据表的增删改
  - DML 数据表里的数据的增删改

1、数据库表管理

1.1、数据库的字段属性(了解)

自增：

- 自动在上一条的记录的基础上+1
- 通常用来设计唯一的主键，必须是整数类型

非空(NOT NULL) 和 空(NULL)：

- 假设设置为 NOT NULL，如果插入数据的时候不赋值就会报错
- NULL，如果不填写值，默认就是NULL

默认值：

- 设置默认的值
- sex，默认为男，如果不指定该列的值，则会有默认的值

扩展知识(阿里巴巴规范)：

- id主键 (主键)

- is_deleted
  - **物理删除(真的从数据库里删除了)**
  - **逻辑删除(用一个字段来表示删没删除，0是删除，1是没删除)**
- create_time 创建时间
- update_time 更新时间

1.2、创建数据库表(create)

```
-- 创建一个school数据库
create database school;
-- 使用创建好的school数据库
USE school;

-- 创建学生表(列，字段)  使用SQL创建
-- 学号INT 姓名VARCHAR(10)，性别ENUM('男'，'女')，身高DECIMAL(3,2)，生日DATE，电话
CHAR(11)，加入时间DATETIME
-- AUTO INCREMENT 自增
-- 字符串使用单引号括起来!
-- 创建字段的部分(12行-20行)，也就是小括号内的语句，除了第20行以外，所有语句后面都需要加逗号
(英文的逗号)!!
CREATE TABLE `student`(
`id` INT NOT NULL AUTO_INCREMENT COMMENT '主键',
-- `id` INT PRIMARY KEY NOT NULL AUTO_INCREMENT COMMENT '主键',
`name` VARCHAR(10) NOT NULL DEFAULT '张三' COMMENT '姓名',
`gender` ENUM('男','女') NOT NULL DEFAULT '男' COMMENT '性别',
`height` DECIMAL(3,2) NOT NULL DEFAULT '2.01' COMMENT '身高',
`birthday` DATE DEFAULT NULL COMMENT '生日',
`tel` CHAR(11) DEFAULT NULL COMMENT '电话',
-- `tel` CHAR(11) UNIQUE DEFAULT NULL COMMENT '电话',
`join_time` DATETIME DEFAULT NULL COMMENT '加入时间',
PRIMARY KEY(`id`),
UNIQUE(`tel`)
)ENGINE=INNODB DEFAULT CHARSET=utf8;
```

```
-- 格式
create table [if not exists] `表名`(
  `字段名` [属性] [索引] [注释],
  `字段名` [属性] [索引] [注释],
  ...
  `字段名` [属性] [索引] [注释]
)[表类型][字符集设置][注释];

-- 查看创建数据表的语句(创建一次，多次使用)
show create table student;

-- 查看数据库表的结构
desc student;
```

1.3、转储(导出)sql文件

1. 右击刚刚创建好的数据库
2. 点击转储SQL文件, 可以看到 `结构和数据` 或 `仅结构` 选项
3. 选择你要转储的类型(`结构和数据` 或 `仅结构`)
4. 选择你要保存的本地路径

5. 点击开始按钮
6. 看到**successfully**表示转储sql文件成功
7. 创建一个新的数据库
8. 右键点击该数据库然后运行这个sql文件, 并查看一下效果

1.4、修改表结构(alter)

> 修改

```
-- 修改表结构
-- 添加字段
-- 新增一个叫做id_card的字段，它的类型是可变字符串且非空
    ALTER TABLE school.student ADD id_card VARCHAR(18) NOT NULL COMMENT '身份证';
-- 修改字段属性
    ALTER TABLE school.student MODIFY id_card VARCHAR(30);
-- 修改字段名(同时也修改属性)
    ALTER TABLE school.student CHANGE id_card id_card1 CHAR(10) NOT NULL;
-- 删除字段
    ALTER TABLE school.student DROP id_card1;
-- 修改表名
    ALTER TABLE school.student RENAME school.stu;
```

1.5、删除表(drop)

> 删除

```
-- 删除表(如果表存在再删除)
DROP TABLE IF EXISTS school.student;
```

所有的创建和删除操作，尽量加上判断以免报错

2、Mysql的数据管理

2.1、外键(了解)

- MySQL外键的作用：
    - 保持数据一致性，主要目的是控制存储在外键表中的数据
    - **外键存在的表叫做从表**
    - **主表**: 字段被其他表引用
    - **从表**: 引用其他表里的字段
    - 删除有外键关系的表的时候，必须要先以从表开始删除，再删除主表

```
-- 员工表的deptno字段要去引用部门表的deptno字段
-- 定义外键KEY
-- 给这个外键添加约束(执行引用)
CREATE TABLE `zhubiao` (
-- CREATE TABLE `dept` (
  `deptno` int NOT NULL,
  `dname` varchar(14) NOT NULL,
  `loc` enum('北京','上海','广州','深圳','武汉','杭州','南京','西安','成都') DEFAULT
NULL,
  PRIMARY KEY (`deptno`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

```
CREATE TABLE `congbiao` (
-- CREATE TABLE `emp` (
  `empno` int NOT NULL COMMENT '员工编号',
  `deptno` int DEFAULT NULL COMMENT '部门编号',
  PRIMARY KEY (`empno`),
  FOREIGN KEY (`deptno`) REFERENCES `zhubiao` (`deptno`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

2.2、DML语言(重要)

2.2.1 添加数据(insert into) 插入

> 语法: INSERT INTO 表名([字段名1，字段名2，字段名3]) VALUES('值1'，'值2'，'值3')...

```
-- 插入时匹配对应的值
INSERT INTO scott.emp(ename,empno,job,mgr,hiredate,sal)
VALUES('华安',9527,'家丁',null,SYSDATE(),300);

-- 不声明字段新增数据，必须把所有字段的值都写上
INSERT INTO scott.emp
VALUES(7501,'刘海柱','法师',7902,'2001-01-01',50,10,40);

-- insert语句后面可以加多个VALUES，可以添加多个数据
INSERT INTO scott.emp(empno,ename,job,mgr,hiredate,sal)
VALUES(9529,'华安1','家丁1',null,SYSDATE(),300),
(7502,'1','法师1',7903,'2001-01-01',50);
```

注意事项:

1. 字段和字段之间使用英文逗号隔开
2. 字段是可以省略的，但是后面的值必须要一一对应
3. 可以同时插入多条数据，VALUES后面的值需要使用逗号隔开即可
4. VALUES，是具体的值，也可以是一个函数

2.2.2 修改数据(update)

> 语法: UPDATE 表名 set 字段名1='修改后的内容' WHERE 字段名2='条件';

```
-- 修改员工的名字华安为唐伯虎
UPDATE emp set ename='唐伯虎' where ename='华安';

-- 将编号为7501的法师，转职为魔导师，工资翻倍
UPDATE emp set job='魔导师',sal=sal*2 where empno=7501;

-- 通过多个条件定位数据
UPDATE emp set ename='华安3' where job='家丁1' and ename='华安1';

-- 修改员工名字,不指定条件的情况下会修改表里面的所有的这个字段的数据
UPDATE emp set ename='华安1';
```

- 条件，筛选的条件，如果没有指定则会修改所有的记录(行)

2.2.3 删除数据(delete)

> DELETE
>
> 语法: DELETE FROM 表名 WHERE 字段名='条件';

```
-- 删除指定数据
delete from emp where empno = 9527;

-- 删除数据(避免这样写，会全部删除)
delete from emp;
```

2.2.4 清空表数据(truncate、delete)

> truncate

作用：完全清空一个数据库表，表的结构不会变

```
-- 清空 student 表
truncate `student`;
```

> TRUNCATE 和 DELETE FROM 的区别

- 相同点：
  - 都能删除数据，都不会删除表结构
  - TRUNCATE 相当于无条件的 DELETE FROM
- 不同：
  - TRUNCATE 会使**自增列的计数器归零**
  - TRUNCATE 不支持事务
  - TRUNCATE 不能删除有外键约束的主表内容
  - DELETE 后面可以添加where条件
  - DELETE 可以回滚**事务**(暂时先了解这个概念, 稍后就会演示一下具体什么是事务)

2.2.5 事务控制

- 概念: 要求多个SQL语句要么同时成功, 要么同时失败, 否则将引发纠纷
- 场景: SMITH想通过银行转账将500元打给KING

> 事务操作演示
>
> 如果开启了事务, 但是不提交, 所做的修改只保存在了客户端, 并没有修改到服务器

```
-- 开启事务模拟真实场景
START TRANSACTION;
-- 从SMITH转账500元给KING
update emp set sal = sal - 500 where ename = 'smith';
update emp set sal = sal + 500 where ename = 'king';
-- 从emp表查询数据
SELECT * FROM emp;
-- 假设此时出现了服务器崩溃的情况
ROLLBACK;
-- 假设一切顺利则提交当前事务
COMMIT;
```

三、Day03 Mysql有趣查询

- **掌握排序和分页的功能**
- **掌握常用函数**
- **掌握分组，过滤和聚合函数的使用**
- **掌握子查询**
- 了解数据库索引的概念

1、排序和分页(掌握)

> 排序(order by)

```
-- 排序
-- 按工资升序排列(从小到大)
SELECT * FROM scott.emp ORDER BY sal ASC;
SELECT * FROM scott.emp ORDER BY sal;
-- 按工资降序排列(从大到小)
SELECT * FROM scott.emp ORDER BY sal DESC;
-- 先按部门升序排列，再按工资降序排列
SELECT * FROM scott.emp ORDER BY deptno ASC, sal DESC;
SELECT * FROM scott.emp ORDER BY deptno, sal DESC;

-- 如果字段的数据都是唯一的，排序为只满足前面的条件
-- SELECT *
-- FROM emp
-- ORDER BY hiredate desc, mgr asc;

-- 想想ename排序(中文的呢要调查一下!?)
SELECT *
FROM scott.emp
ORDER BY ename;
```

> 分页(limit)限制

```
-- 分页查询
-- 为什么要分页？ 缓解数据库压力，给人的体验更好
-- 显示工资前五的员工信息
    SELECT *
    FROM emp
    ORDER BY sal DESC
    LIMIT 0,5;
-- 显示工资6-10名的员工信息
```

> **语法：LIMIT x,y**

- x代表**起始下标**
- y代表**显示多少个**

2、函数

2.1、常用函数(了解)

```
-- 数学运算
SELECT CEILING(9.4) -- 向上取整
SELECT FLOOR(9.4) -- 向下取整
SELECT ROUND(3.1415926,2) -- 保留2位
```

```sql
SELECT rand()  -- 生成一个随机数

-- 字符串函数
SELECT CHAR_LENGTH('你好') -- 字符串长度
SELECT CONCAT('你','好') -- 字符串拼接
SELECT LOWER('HELLO') -- 字符串转换成小写
SELECT UPPER('hello') -- 字符串转换成大写

-- 时间和日期函数()
SELECT SYSDATE() -- 获取日期时间
```

2.2、聚合函数|分组聚集(常用)(掌握)

- 用作统计使用

函数名称	描述
SUM() sum()	求和
AVG() avg()	平均值
MAX() max()	最大值
MIN() min()	最小值
COUNT() count()	计数

```sql
-- 简单使用
SELECT SUM(sal) AS '总和' FROM emp;
SELECT AVG(sal) AS '平均值' FROM emp;
SELECT MAX(sal) AS '最大值' FROM emp;
SELECT MIN(sal) AS '最小值' FROM emp;
SELECT COUNT(sal) AS '计数' FROM emp;

-- 同时查询多个聚合函数
SELECT
    SUM(sal) 总和,
    AVG(sal) 平均值,
    MAX(sal) 最大值,
    MIN(sal) 最小值,
    COUNT(sal) 计数
FROM
    emp;

-- 聚合函数支持哪些类型
SELECT SUM(ename), AVG(ename)
FROM emp;

SELECT SUM(hiredate), AVG(hiredate)
FROM emp;

SELECT MAX(ename), MIN(ename)
FROM emp;

SELECT MAX(hiredate), MIN(hiredate)
FROM emp;
```

```sql
-- 关于Count的计数
SELECT COUNT(ename)
FROM emp;
SELECT COUNT(comm)
FROM emp;
-- Count(指定字段)，会忽略所有的null值
SELECT COUNT(comm) from emp;
-- Count(*) 则显示总共的行数，不会忽略null值
SELECT COUNT(*) from emp;
```

> 聚合函数练习

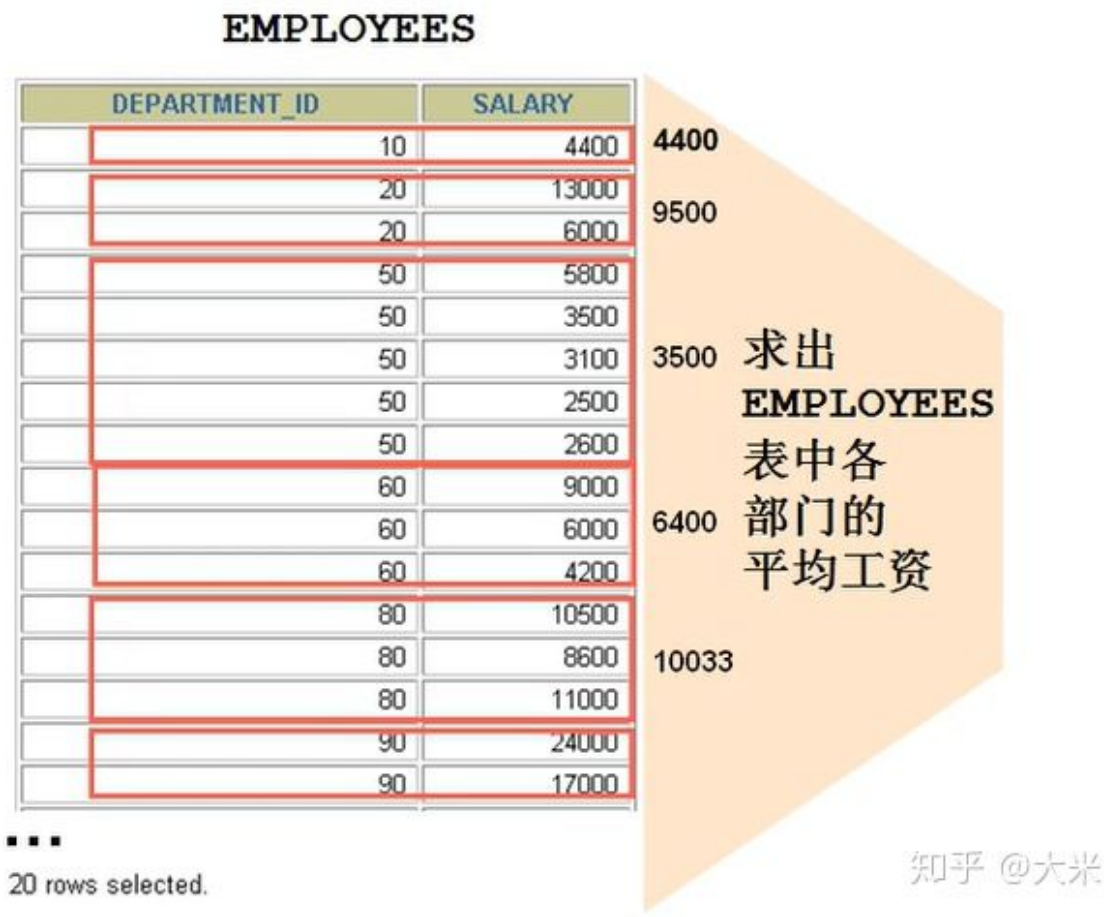
```sql
-- 查询公司员工工资的最大值，最小值，平均值，总和
SELECT max(sal), min(sal), avg(sal), sum(sal)
FROM scott.emp;

-- 查询部门编号为30的员工个数
-- 要用到where判断条件
-- 要用到count函数
SELECT count(*)
FROM scott.emp
WHERE deptno = 30;

-- 查询员工表中的最大入职时间和最小入职时间的相差天数，以different命别名
-- 提示：可使用DATEDIFF()函数，然后这个函数有两个参数
-- 该函数只能作用在日期的计算上
SELECT datediff(max(hiredate), min(hiredate)) AS 'different'
FROM scott.emp;
```

2.3、分组查询和过滤(掌握)

```sql
-- 查出所有员工的平均工资是下面这样写
SELECT AVG(sal)
FROM emp;
-- 那么如果我想根据部门来查看平均工资，该怎么写呢?
```



- 想查看不同部门的人的平均工资, 可以使用GROUP BY语句

```
-- GROUP BY 字句语法
SELECT 列(要求出现在GROUP BY的后面)，分组函数
FROM 表
[WHERE 筛选条件]
GROUP BY 列(将被分组的字段)

-- 注意：查询列表比较特殊，要求是分组函数和group by后出现的字段
```

```
-- 简单的分组查询
-- 查询每个岗位的平均工资
SELECT job, AVG(sal)
FROM scott.emp
GROUP BY job;

-- 查出每个部门的人数
SELECT deptno, COUNT(*)
FROM emp
GROUP BY deptno;

-- 查出每个岗位的最高工资
SELECT job, MAX(sal)
FROM emp
GROUP BY job;

-- 查询每个部门中每个岗位的人数和平均工资
SELECT deptno, job, count(*), AVG(sal)
FROM scott.emp
GROUP BY deptno, job;
```

```sql
-- 分组前查询
-- 通过岗位进行分组后，查询员工名称带字母E的人有多少，并显示有哪些人
SELECT job, count(*), group_concat(ename)
FROM scott.emp
WHERE ename like '%E%'
GROUP BY job;

-- 查询除财务部门以外的其他部门之中的最高薪资，最低薪资和平均薪资分别是多少
SELECT dname, MAX(sal), MIN(sal), AVG(sal)
from dept d
INNER JOIN emp e
on d.deptno=e.deptno
where dname != '财务'
GROUP BY dname;

-- 分组后查询
-- 查询哪个部门的员工人数<6
SELECT deptno, COUNT(*)
FROM emp
GROUP BY deptno
HAVING COUNT(*) < 6

-- 查询各岗位中员工名称包含字母E的员工人数，并且只显示超过2人的数据
SELECT job, COUNT(*)
FROM scott.emp
WHERE ename like '%E%'
GROUP BY job
HAVING COUNT(*)>2;

-- 分组后查询，查询不同部门名称的
-- 最高薪资，最低薪资和平均薪资
-- having 条件:平均薪资大于900
SELECT dname, MAX(sal), MIN(sal), AVG(sal)
FROM dept d
INNER JOIN emp e
ON d.deptno=e.deptno
GROUP BY dname
HAVING AVG(sal) > 900;

-- 查询除了销售部门外的其他部门中的最高薪资，最低薪资，平均薪资分别是哪些
-- 并且只显示最低薪资大于900的数据
SELECT dname, max(sal), min(sal), avg(sal), GROUP_CONCAT(ename)
FROM emp e
INNER JOIN dept d
ON e.deptno = d.deptno
WHERE d.dname != '销售'
GROUP BY dname
HAVING min(sal)>900;

-- 总结:
  -- 聚合函数COUNT、SUM、AVG、MAX、MIN等不能在WHERE子句中使用，只能在分组后配合Having语句使用
  -- 普通函数，比如: ceiling(), floor(), rand()可以在WHERE子句中使用
  -- WHERE语句先执行过滤，HAVING语句后执行过滤
```

3、子查询(掌握)

- 目的：子查询适合于作为查询的筛选条件
- 本质：在WHERE语句中嵌套一个查询语句

```
-- 子查询又称作嵌套查询
-- 执行顺序由里到外
-- 子查询返回一个值作为外面的WHERE语句的条件

-- 查询销售的所有员工(员工编号，员工姓名)
SELECT empno, ename
FROM emp
WHERE deptno IN (SELECT deptno FROM dept WHERE dname='销售');

-- 查询在广州工作的员工中，哪些人的名字中包含有E，要求显示员工姓名和部门编号
SELECT ename, deptno
FROM emp
WHERE deptno IN (SELECT deptno FROM dept WHERE loc = '广州')
AND ename like '%E%';

-- 查询SMITH的领导姓名
SELECT ename
FROM scott.emp
WHERE empno IN (SELECT mgr FROM scott.emp WHERE ename='SMITH');

-- 查询编号是7521,7566,7654,7698,7782的员工，哪些是BLAKE的下属
SELECT *
FROM emp
WHERE empno IN (7521,7566,7654,7698,7782)
AND mgr IN (SELECT empno from emp WHERE ename = 'blake');

-- 查询KING的下属哪位在上海工作，要求显示员工编号，员工姓名，岗位，领导工号，部门编号
SELECT empno, ename, job, mgr, deptno
FROM emp
WHERE mgr IN (SELECT empno from emp WHERE ename = 'king')
AND deptno IN (SELECT deptno from dept WHERE loc = '上海');

-- 子查询返回多个值
-- 查询北京和上海的所有员工信息
SELECT *
FROM scott.emp
WHERE deptno IN (SELECT deptno FROM scott.dept WHERE loc IN ('北京','上海'));

-- 查询KING的工资占公司总工资的比例
SELECT sal/(SELECT sum(sal) FROM scott.emp)
FROM scott.emp
WHERE ename='KING';
```

4、索引(了解)

- 作用: 通过索引可以让我们更快的获取数据库里面的结果

4.1、索引的分类(了解)

- **主键索引 (PRIMARY KEY)**
  - **唯一**的标识，**主键不可重复且不能为空**，大多数情况下只能有一个列作为主键索引
  - 同时设置UNIQUE 约束和NOT NULL约束
- **唯一性索引 (UNIQUE KEY)**

    - 唯一的标识，唯一性索引**不可重复但是可以为空**
    - 避免重复的数据出现，唯一索引不可以重复，多个列都可以标识为唯一索引
    - UNIQUE 可空，可以在一个表里的一个或多个字段定义
- **常规索引 (INDEK/KEY)**
    - INDEX关键字来设置, 让查询语句可以执行的更快
- **索引语法参考**
    - 添加索引
        - `ALTER TABLE 表名称 DROP INDEX 索引名称`
    - 删除索引
        - `DROP INDEX 索引名称 ON 表名称`

4.2、索引原则(了解)

- 索引不是越多越好, 一般来说数据要500w以上才考虑是否要加索引
- 不要对经常变动的数据加索引
- 小数据量的表不需要加索引
- 索引一般加在常用来查询的字段上

5、欢乐写sql

```
-- 查询姓名是SMITH员工他的领导姓名
-- 子查询找到smith
-- 用子查询的结果去找到领导的名字


-- 查询北京和上海的员工信息
-- 子查询找到北京和上海的deptno
-- 然后去emp表通过子查询的结果找到对应的员工


-- 查询在广州工作的员工中，哪些人的名字中包含有E，要求显示员工姓名和部门编号
-- 首先来控制显示员工姓名和部门编号
-- 然后找到广州工作的员工，需要用到子查询作为条件1
-- 接着将名字中包含E的人查找出来作为条件2
-- 条件1 and 条件2


-- 查询编号是7521,7566,7654,7698,7782的员工，哪些是BLAKE的下属
-- 首先通过in关键字找到上述的员工作为条件1
-- 然后用子查询找到blake的empno编号做为mgr使用作为条件2
-- 条件1 and 条件2


-- 查询KING的下属哪位在上海工作，要求显示员工编号，员工姓名，岗位，领导工号，部门编号
-- 查找king的下属，mgr=king的员工
-- deptno要在上海
-- 显示具体内容


-- 给在'北京'工作的员工，工资增加0.01元
-- update emp_copy set sal=sal+0.01 where deptno = 北京的员工deptno;


-- 获取一个随机的整数
```

```sql
-- select rand()
-- select round(rand()*100, 0);
-- ceiling 向上取整
-- floor 向下取整


-- 把姓名为ADAMS的数据删除掉
-- delete from emp where ename = 'ADAMS';


-- 查询每个部门的人数，只显示4人以上的部门
-- 首先是显示部门的人数，那么需要显示deptno，以及count(*)
-- 然后还需要group by分组一下
-- 最后需要通过having来显示4个人以上的


-- 查询在'上海'工作的员工每个岗位的人数，要求只显示超过1人的岗位，按人数升序排列


-- 尝试创建一个数据表
    -- 在任意数据库创建一个`emp_new`数据表
    -- 数据表包含的字段有`id`,`name`,`birthday`,`tel`
    -- 字段属性可自行定义
    -- `id`拥有主键索引，`tel`拥有唯一索引
-- 尝试过后可参考下方的示例进行复盘!
-- CREATE TABLE `emp_new`(
-- `id` INT AUTO_INCREMENT NOT NULL COMMENT 'ID',
-- `name` VARCHAR(5) NOT NULL COMMENT '名字',
-- `birthday` DATE DEFAULT NULL COMMENT '生日',
-- `tel` CHAR(11) DEFAULT NULL COMMENT '电话',
-- PRIMARY KEY(`id`),
-- UNIQUE KEY(`tel`)
-- )ENGINE=InnoDB DEFAULT CHARSET=utf8;


-- 尝试给数据表新增一个字段
    -- 往`emp_new`数据表新增一个字段id_card，整型，非空，注释为身份证
    -- 尝试过后可参考下方的示例进行复盘!
-- ALTER TABLE emp ADD id_card INT NOT NULL COMMENT '身份证';
```